# LEONARD KLEINROCK INTERVIEW

There's a key idea out of information theory which I started to talk to you about which says that if you take -- whereas individual users or data streams may behave very unpredictably, a collection of a large number of them behaves very predictably. And one understands that with a bursty stream, such as a data stream, you don't want to have a circuit switch where you keep the connection up all the time. You want to be able to let go of the connection or the resources when you're not using them, because you tend to use them in bursts. But the other half of that story is the result I just told you, is (if) you have a large number of them, they behave in a very predictable way. So I can tell you just how many people I can support on one communications link if you tell me the bandwidth of the link and the bandwidth requirements of each user, even though each user's unpredictable. And it's something you can calculate as well. Without being condescending, if I told you I had 100 users each using 5% of a communications line on average, highly bursty, okay? So you'll (lease) the line 5% of the time. And if you have 100 users, how many lines do you need? The answer is you need five lines, because each user uses essentially 1/20th of the line.

I see.

Now that's a calculation without allowing for statistics. And the point is the unpredictability goes away when you gang up a bunch of users independently. I didn't say that very well. Another way to say it is this. Let's get to gambling. If you toss a coin, say half the times it's heads, half the times tails. You toss if five times and you get say four heads and one tails. You're not surprised. Even though you expected like well, say six times you tossed it. You split the three heads, three tails. Right? Make it five heads and one tail. If you toss it a million times, you'd better get almost exactly a half a million heads and a half a million tails, and if you got like 480,000 heads and 520,000 tails, that coin is not fair. Guaranteed. The likelihood of that being fair is one in billion, billion, billion. You know, I can give you the numbers if you like. And the way to characterize that is to say that if you let nature take a good crack at you, she's going to expose her average behavior to you.

Sounds like something Shannon would say.

In fact, that's almost Shannon's words. I took that notion and applied it to the world of packet switching -- of data switching -- and said, *"Look, if we got these --."*

I'm sorry. But where are we chronologically now?

**Chronologically is when I started working on my dissertation.**

And that would be what year?

**I submitted the proposal sometime in 1959. I forget when. I think the fall. You know, I came in the fall of '58 and worked for about a year in the Chess ____ Program and then I went into networking stuff. And so I began to investigate the behavior of message switch networks. And I started to tell you the difference between message switching and package switching. In message switching, you take the entire message and you transmit it as a bulk, hop, hop, hop. And the trouble there is if the message is large, each time you transmit it over a communications line, at the other end, you've got to wait until it all comes in before you start relaying it over the next hop. If the message is very long, it's going to take a long time for each hop and many hops and you've got a long delay. And that's the way the teletype network was working and a variety of others in those days. I analyzed the behavior and was able to show that the response time of these things was dependent upon the life of the message, and if you could use shorter messages, you'd have a much better response time. Shorter messages means break them up into little pieces called packets. We didn't call it packet switching. I was able to show for analysis the effect of the size of the block you were transmitting and what it meant to the overall delay. And in particular, nobody had been considering queuing effects. Everybody typically assumed there was nothing in your way when you went hop, hop. But each node you got to wait as well. And if you got to wait for other long guys, it magnifies the effect of the size of the message.**

Sure.

**Okay. So that was built in and all of that came out in the beginnings of the theory I was developing. So what I did is I analyzed --**

I'm having trouble making the connection. How did you go from this whole idea of, you know, average behavior --

**Oh, how did that get into the picture?**

-- into data switching. It isn't making sense to me.

**Oh, hit the point. You got a communications line. That's the good of bursty stream.**

structure of the network that's supporting that kind of traffic? And so I analyzed a number of things. I started out -- first of all, one had to make a model of this. And I developed an analytic model that very well captured the behavior of the real network. There were some very important things I had to do. Mathematically you can't solve these networks. Never mind design. But if I give you a network, you can't analyze it. They're just too hard. For reasons I can explain. You probably don't want to know. But there were some essential problems. So I had to introduce some assumptions and some approximations which were exactly the right approximations to make the model yield to analysis and still represent the real world very accurately. The key assumption is something called the independence assumption. Do you care about it?

No.

All right. So the point is I developed a workable analytic model for the performance of data networks. I then went into some design issues. Decided how you should lay out a network, how you should decide how fat the line should be, and how you should move the traffic through the network -- that's called routing. And I developed and invented some algorithms. Analyzed them and optimized them. The optimization, of course, was the design step. Now by the way, in that same work -- you understand nobody was interested in this stuff at that time. Nobody. This was a piece of work that I was doing because I was interested in it. I had to invent some of my own tools. I never studied queuing theory. I had to develop some tools of queuing theory and take what was known, because this was a problem in delay which is what queuing theory is all about. In fact, nobody at that time had used queuing theory to evaluate computer systems at all. And so a second compliment of my work, besides that in networks, was in the other really exciting development of that time, which was timesharing. Timesharing was just happening in the very early '60's when I was into this. Okay? '59 to '62.

Uh-huh.

In fact, one of the very first timesharing systems developed was at MIT.

Right. Yes, I know.

And so I quickly took that and I developed a model for it. The first analytic papers on timeshared modeling was in my thesis as well, and that was a whole other development that was basically developed in the 60's, which you probably don't care about.

Well, I care about it in so far as timesharing's effect on networking. The relationship between timesharing and networking.

Okay. Well, it's a good point. It's one of the reasons we had to develop a network in Arpa, and I'll explain that in a minute. Anyway, you have to understand the common element here was queuing theory. Queuing theory asks the following questions. What's the (proof) of the system? By the way, I'm going to couch these questions using terminology that makes sense to networking, as opposed to makes sense to somebody studying ____ of the processes. The mathematician doesn't use these words. Mathematician talks about other things. _____. Capacity. How much can you support? Response time. Buffer size. All the systems elements that you need to concern yourself with when you're building something like a network or like a timesharing system. How many users can you support? How long will it take me to get a response? How long will it take the message to get through the network? How fat must be the channel here? How much buffer storage do you need in the switch? So these are the things I was concerned about and the obvious analytical tool was queuing theory.

A second tool, by the way, was called network flow theory. And that's, by the way, what Howie Frank was involved with, and we'll get to that in a minute. But you can address some of the network flow problems through queuing theory and I'll tell you what the overlap was later. Now in addition to developing these tools and these fundamental principles of data networking, I did this monster simulation. There's where the couple at Lincoln Lab was on _____. They had this wonderful machine called TX-2, which I could get my hands on, and you know, the summers I was there, and I was there with Larry and those other guys, and Wes Clark, and all the guys. You probably didn't hear names like Froggy and Stocky Brand and Peterson. All the designers of the TX-2.

Describe the TX-2 to me.

Ah, [smacks lips]! It was a jewel. You have to understand. It was a machine in development while we were using it. It was the second transistorized machine built at Lincoln Lab. It was well ahead of its day. It had instructions that were not used in the large machines for like a decade or two later. With one instruction, I could -- first of all, it had a 36-bit world, which is large, and you could break it into four sections of nine and do four operations independently on these nine-bit segments, or you could break it into one 18 and two nines or two 18's, as you like. With one instruction I could test a bit to see if it was a one or a zero, and based on that test, I could either change it or not, set it to a one or not, and rotate the whole word

around one bit. So if I was looking at say the least significant position -- do you know what I'm talking about? Here's 36 bits lined up. Like 36 integers, the least significant integer is the one's position. Well, similarly in the digital world. If you look at that bit, you do all the examination and testing of it. When you're done with that, all on one instruction, you could rotate the word around so the second digit appeared in the first position, and you keep testing it. So it's a very efficient machine. Anyways, it was very advanced. Let me just put it that way. It was built by a collection of hardware designers and logic designers, who themselves were the programmers. You see, this was a machine where you were not a specialist, but you did it all, and these guys were super good.

And Larry was involved with it?

He was not one of the designers. He came in later after the machine was built. But he wrote the second (compiler). It was called --

The story you were telling last night about how he came in and wrote.

Yes. It was amazing what he did. I mean. I'm trying to remember the name of the compiler he wrote. I want to say Meta, but that wasn't his. I'll think of it. I still have my program, by the way. But the first program I ever wrote for that machine, you know, I was a neophyte programmer. The way you wrote programs for that machine, you'd have this big console there. Okay? Big room with this machine. People working on different pieces of it. And you'd sign up for hours at a time or half an hour at a time. You sit down there and there was a bank of little toggle switches and you'd set the ones and zeros in the toggle switches, which was your program or possibly you'd put it on a paper tape. This is no Fortran. This is called machine language with a little bit of assembly capability. Anyway, so I sat down and tried to run a program and it wouldn't run. You know, these things just stop immediately and Peterson came by. I forget his first name. He was one of these guys who had been designers. He was an expert programmer. And he watched me and he said, *"Let me help you."* He said, *"Program's fine. Just needs a little bit of tuning."* So he tuned a little bit and choo!

Really?

I mean, it was like artistry. You know, these guys they really understood.

And you hadn't done much programming?

I hadn't done any programming before that.

So it was a whole new world.

Oh, it was unbelievable. I was suddenly thrust in with these guys at Lincoln. These were professional guys. They knew how to build paper tapes. Punch paper machines. The first of Xerox's Xerox machines, I believe, was there. It was a huge machine with a big drum and it spit out paper that wide and it was a continuous role, and we had these big scissors, and when you turned that printer on, you had better go over there and start cutting. If you fell behind, it was like the Sorcerer's Apprentice. It would be all over the place.

So what was it like to get introduce to computers.

You never got introduced formally. You just were thrown in this group of people who were using computers and you get some documentation about the logic design, about the instruction set, about the functional description of the machine, and you'd be around these guys and you'd ask questions, and you'd look at some literature and you'd watch them and you'd start doing things. I had a job. You know, the first guy I worked with at Lincoln Lab was Ken Olsen.

Oh, right. That's in your _____. You were building some circuit boards?

Well, I did a few things. First thing he wanted me to do was to study the behavior of gamma rays on flipflops. A flipflop is a basic element of a register. So we started doing that. And then he had me design something called a variable pulse delay unit. You hit the clock here and some fixed amount of time later, you want a pulse to come out, and that's variable. He wanted me to build it. I did. And he kept having me improve it here and there. He was very gentle, you know. He says, *"That's good. But how about trying to do this."*

Now, I thought he was in Wes's group.

You know, I don't know what the management structure was then, because it was my first summer there, '57, and at the end of the summer, he left. Yes, he was in the TX-2 group, but I don't know who was in charge at that time, to be honest with you.

And that was Wes. Wes was at TX-2.

Yeah. Yeah. Wes was I think one of the _____. He was the manager of TX-2 development. At any rate, then Ken left, and I did not go with him, of course, and I'm happy about that. But he took a lot of the good designers with him, by the way.

Yeah, I've heard that.

Anyway, where was I? What was the environment like? I got to my simulation. That's where we entered the TX-2 thing. That machine was available to me, because I was an employee and I was able to get that machine from midnight to 7:00AM, a huge shift, four days a week for a period of three months. This was toward the end of my dissertation. In order to debug my (Mansa) simulation. Mansa was 2500 lines of code. Trouble is those four days were not contiguous days. So it would be like Monday, Tuesday, Thursday, Saturday, or something. So you can imagine what happened with my sleep habits. You probably remember on the video clip the story about Larry Roberts and the way he scared the hell out of me one night?

All right. Tell me that story again. You were a little dramatic with it. Tell it to me just in plainer --

Well, what happened was, you know, I'd be there. You know, I'd arrive. I'd be working all day pretty much and midnight came and I'd take the machine over. And you know, people were gone by then so I was all by myself with this wonderful machine with all these parts which could break down and, you know, it was a million dollar machine, which to a kid like me was very important. And there were always pieces missing that were under repair. The registers -- the panels on the console were about six inches and about an inch and a quarter high, and these were the kinds of modules that plugged in, and there were little LED's that would light up and (all you might expect). And you'd get to know the machine like your body, you know. In fact, they had an audio -- a speaker on one of the registers in the CPU, and when it went up and down, it would make a noise, and when you set it running on the program, it would go [makes noises], and that was good. It was doing things. And if it ever went [whistles] it meant it was in a loop and you've been had. So you got to know. You heard what the paper tape reader sounded like and there was air, a cool fan here, and you knew what that was, and if anything was wrong, you'd hear it immediately and it would scare the hell out of you, because you don't want something to break, because that's an indication maybe something's crashing. So one night I'm there at about three in the morning and you really get tired and grumpy and you go into a trance, because half the time you're waiting for the thing to compile, so you're doing nothing. And I heard this funny sound. This s-s-s-s-s sound and right away my adrenalin started going *what is wrong here?* And it is dramatic.

**And I looked around and in one of the empty spaces -- one of these little modules was missing -- I saw this pair of eyes looking out. Larry had snuck into the room and tried to scare me and he succeeded. I could have killed him!**

That's wonderful. So was that like Larry to play a joke on you?

**Not too much. That was especially amusing. He and I were good buddies. We did a lot of weird things together. Lots. I can't tell you all of them.**

Really? Well, you've told me about the Colorado River. I guess that was weird.

**Yeah, but no. We did some really strange things.**

Just things young people do?

**Yeah, you know, mischievous things.**

Oh, mischievous.

**Oh, yeah.**

Anything almost innocuous you could tell me about?

**No. The gambling (in Silva) are exciting enough. At any rate, typically on those nights, come 7:00AM, one of my classmates would come in. I don't remember who it was exactly. See my thesis supervisor graduated five students. First was Erwin Jacobs, Ed Hoffstetter, Jack Rosenfeld, then myself, and Hersch Lumas. They came in two, one, two, and after that he left. Now Hersch Lumas was with me and he would use the TX-2 as well. He'd come in at 7:00AM typically bright-eyed, bushy tailed, well-shaven, smelling good, and I'd be, you know, drinking coffee and bearded and dirty, and I'd look at him and say, *"Get out of here!"* You know, do the contrast and then I'd go home. And I'd always try to get home before the sun came up, because if the sun came up, I knew I had lost the night and I just wanted to get into bed before the sun rose, and I often didn't make it. Once I went home in a snow storm, just freshly snowed, and I really wanted to get home. Had to climb a hill to get home and the car was slipping, _____, and I just kept -- these were new tires -- I kept gunning and gunning, and I ruined a full set of new tires getting up the hill.**

And you were a poor graduate student.

**Yeah. Yeah. *New* means I bought them in a junkyard.**

Right. Okay, we're getting off a little bit. So you were --

**So I did this big simulation. The one thing that was kind of a semi-heroic thing, when I wrote my program in simulation, I wrote the full simulation before I debugged any of it. I just chose to do it that way. That's a very bad programming practice in general, but I like to take these challenges on. And if that simulation program didn't work, I was in danger of not getting a PhD.**

Really?

**Well, I had to prove my approximations were good. I mean, you know, you can assume anything. You had to prove them. And I finally did get it working, fortunately, in that massive effort over this period of three or four months, and it proved out my theory and that was a big success. But the TX-2 -- without that machine, I wouldn't have been able to take on that ambitious kind of simulation. It was highly interactive. I had things you don't even have now on machines. We had light buttons on the machine. It was a little pointer and you could point it with a light pen. Light pen was a photoelectric device connected to the computer and if you point it to -- for example, suppose it had this little thing as a mosaic. If I point to it, then the pen only sees that light when the screen activates it. So the pen knows where that is because it knows when it saw that and we have _____ and timed position. So if I hit that button, it means the one thing and this button means another. It was a highly interactive program. You could stop it. You could see the histograms growing as the data came in. You could see the traffic moving. And you got the analytic results as well -- the numeric results. Anyway, I still have that program and, in fact, I have a piece of paper which indicates when they said, *"The TX-2 is being decommissioned."* Which meant that program would never again run in any machine.**

Really? Tell me about the TX-2 SDC experiment. Do you know --

**That came later.**

That was in what? '62?

**I believe it was '62 or '63.**

Do you know much about it?

**No. Just I didn't see it happen. I just know how Larry reported it.**

So I should really ask Larry more about that.

**Yes. That for sure. Not me. But let's go to the timesharing thing. Just go off on that tangent for a minute. Timesharing came into its heyday in the early '60's for a very good reason, by the way. The early use of computers, or the way I describe it, you got to sit down (on council) and get free use of the machine. Not free. No. Full use. But then when the IBM world came on, they went to batch processing, and now you would take two or three days to do the smallest little thing to discover one error in your code, and so the turnaround time was awful. So timesharing was invented to solve that problem. Where you can get rapid response because more than one user could be using the machine at the same time. So these timesharing machines became popular and then Arpa, of course, began to support Computer Science research. In the early 60's, one of the main things they supported was timesharing. I was doing a lot of analytical work then deriving new ways to use timesharing systems. I invented something called processor sharing and I analyzed round robin. I invented a whole bunch of algorithms.**

Now Liclighter had a lot to do with timesharing.

**Liclighter. Yeah. Timesharing. They set up the Arpa office, of course. But he didn't do any analysis. Okay? He was just supporting that kind of work.**

Right.

**As a result, I told you about the PDP-10 problem, okay? Well, before the network was there, these machines were being deployed out of the network. So you're a researcher at the University of Utah and Arpa comes and agrees to support your research. You're going to say, *"Listen, buy me a computer so I can do research." "Buy you a computer. Fine."* So a lot of these timesharing systems were sprinkled around the research community, and every time a new researcher came in, they wanted to get their own machine. Trouble is each of the machines that they put out there began to develop in a unique fashion. Once you give it to a bunch of smart guys, they're going to do things with it. Suddenly, there's a unique resource at each of these facilities. So how do you make all of that resource available? You can't duplicate it everywhere. So the notion of needing a network came on. That was one of the prime motivations of the network.**

Well, this is what Bob Taylor talks about.

It served a military use. Okay? But in order to support his own research community he wanted to make the Iliac 4 and the simulation work at UCLA and the graphics work at Utah available to everybody.

This is Taylor?

Taylor and Larry. So resource sharing is quoted corrected to be the prime mover for generating the network.

Sure.

Larry basically used that argument to justify the economics of the network. He said if we had to replicate this --

Yeah, I know.

You know. Now that's an example of Larry gilding a lily. Nobody is going to put an Iliad everywhere. But he was effective. He would go to Congress and the Budget Committee and say, *"Listen, it will cost you "X" million to do this and a fraction of that with the network, and the network's only costing you so much. It's a good deal."* Larry will exaggerate and to good effect, you know. Okay. So now where are we? I'm finishing up my dissertation. It's very well received at MIT. In fact, it's so good they decided to make it into that McGraw-Hill book series from Lincoln Lab. The book that you've got a copy of. Only the best works have made it to the book and I was very flattered when they said they wanted to do that. I was surprised at that. Because again, no one was really interested in this area. So it was on the analytic and technical strength that it did that. Not on the application. I remember Peter Alias was on the committee where I had to present this final result. I don't know if you know him. He's one of the great information theorists, and Shannon, and (Arthurs), and a queuing theory type from the management school again, Galloho. What was his first name? I want to say Ed, but anyway.

Did you consider yourself a queuing theorist at the time?

No. Engineer. Engineer who was using queuing theory. As it turns out, in my thesis, I developed some brand new stuff in queuing theory. Something called a *conservation law*, which is still being generalized to this day. So I did some fundamental work with queuing theory, but I didn't do it for the purpose of queuing theory. I did it because I wanted to develop these engineering systems.

Is the conservation law relevant to anything?

Yes. Certainly to timesharing and, in fact, to networking these days. Here's what it is. Ready?

Uh-huh.

There's a thing called priority queuing where some groups have to be treated better than others. Generals versus privates. Okay? Short messages versus long messages. So you have a whole bunch of priority groups. Now it stands to reason that if you help one group, which means you give them preferential treatment, you're going to hurt another group. The question is what's that balance? If I reduce your waiting time by two seconds, do I increase his waiting time by two seconds? And I characterize exactly what goes on there.

Interesting.

I said there's a certain conservation of something which is always the case, no matter what the algorithm is. No matter how you balance people off, you can't ever beat this. It's like the (Heizenburg) principle. You're going to have that much. And it was very important in showing that -- don't ever try to beat it, because you can't, and it's a good test of when you have consistent proof of the way an algorithm looks, etc. And it's been generalized considerably since that work, which is nice. Again, I did the first model of timesharing, as I said, which launched the whole field of performance evaluation. But the answer is no. I never considered myself a mathematician type queuing theorist. In fact, however, there was an important contribution here in the following sense. You can take this for what it's worth. Queuing theory was begun by an engineer, a guy named (Airlang), in the early part of the century, and he developed lots and lots of stuff. He was an engineer. He was solving telephone engineering problems. Okay? He was my kind of guy. In the early 30's, in fact, in 1928, a book was written based on his work which publicized his work. The mathematicians took note of it then because they saw this book come out. In fact, William Feller -- you may know Feller, F-e-l-l-e-r.

Huh-uh.

He wrote the key books on probability theory. Your dad would have known him or your grandfather.

Oh, I'm sure.

Anyway, Feller was an expert in something called a birth/death process. The point is Feller noticed that what Airlang was talking about was a birth/death process for which Feller had all these results. So Feller coupled the two and published all this, and the world of mathematics suddenly captured queuing theory and they took it off into that direction, mainly into the mathematics world. They worried about the things mathematicians worry about, esoteric stuff. Meanwhile, queuing theory was lost to the engineer in a sense.

Then came World War II. World War II was new field called Operation Research which got developed. Operation Research was simply a collection of other systems engineering topics like search theory, gain theory, linear programming, queuing theory. All these things which they hoped would solve like how do you move men from here to here. People thought that Operation Research was going to solve major engineering problems at the end of the war. End of the war came by and people started using it. They had developed all these Operation Research groups and it fizzled, because the models were too simple. So Operation Research was sort of a dud.

This was in the 50's. In the 60's, timesharing happened. In the 50's, computers happened. 60's, timesharing happened. And I came in and I showed the world that, look, queuing theory is just the right tool to evaluate this wonderful phenomena called timesharing. So people got real excited. And in the 60's as well, and particularly the early 70's when computer networks came out, I had developed the theory, but I showed the same theory can be used to evaluate data networks. That became popular. So again, queuing theory just blew up. So if you ask if I'm a queuing theorist? No. But did I do something to the queuing theory besides my little contribution of the conservation law? Yeah. I shifted its focus, so that queuing theorists now think about these applications when they do their work. In fact, the entire European community which studied only telephone problems. Remember, 1967, I went to my first meeting of what's called the International Teletraffic Congress, ITC, and I presented a paper on modeling timesharing and it turned their eyes around. And then a couple of years later, I did one on networking. And so all those grand old queuing theorists and the telephone engineers started to need data.

So tell me about the application of queuing theory to networking. What is all that?

Okay. People were unaware that there was this magnification effect of delay when you began to load a system down. In fact, IBM committed a horrible error when they came up with their first timesharing system. They thought you could run the utilization right up to 100%. Use the CP 100% of the time. And until you got past

that point things would be fine. But in fact, things get worse and worse as you get closer to 100% because of the time you spend in the que, which they didn't account for. The same thing in networking. When you start loading that align. You spend not only time in transmission, but time in the buffers waiting to get your turn. And that very much affects the way you design networks. It very much affects how big a buffer you need. It affects where you send traffic. Try not to bring them all to one place necessarily if the capacity is not there.

So, for example, one of the things I did in my network design was the following. This has to do with the adaptive routing again, which we can touch on that if you like. My results showed that if you can take all the traffic, send it down one line, and get that line an enormous capacity, instead of breaking traffic into little filamentary pieces and sending it down many small capacity lines, there's a big one. It's called the economy of scale. Analytically, sure. There's a great gain. In fact, if I can double the traffic and double the capacity, I will have the delay. And if I make that factor 100, then I drop the delay by a factor of 100. This was basically unknown to engineers and to queuing theorists. It was unknown to engineers because they'd never seen it before. Unknown to queuing theorist because they didn't care. Your queuing theorist just gives you the equation for behavior and never looks at the trade off. That's an engineer's job.

What did that to routing? If you could, you'd like to design a network where you had a lot of fat channels with a lot of traffic on those fat channels. Trouble is that assumes you know what the traffic's going to be. You know how much traffic's going to go from one node to another node, from UCLA to MIT, and if you knew that, you could design a perfect network. Trouble is you don't know the traffic records ahead of time. And even if you did know, it changes in time. So which traffic makers do you design for? So what do you do? Well, let's look what alternate routing or dynamic routing does. Dynamic routing or alternate routing says you've got traffic coming in here and there's alternate routes. Well, there's usually a primary route and an alternate route and maybe more than one alternate. Well, the primary route is usually faster and better and shorter. You only take your alternate route when this gets busy. So from the point of delay, alternate routes may be bad because they're longer. Secondly, when you take the traffic and you do that to it, that's not collecting it, which is what my theory said. My theory said, *"Don't do alternate routing."* Except I recognize also that you don't know what the traffic's going to be. So here comes two philosophies.

[Telephone rings. Tape is turned off.]

Well, there's two view. One view is if you know the traffic, you could design a perfect network to match the network design to that known traffic. That's what the theory says. The practicality is you don't know the traffic report, traffic matrix. So what you do is you design -- you allow the network -- you design some network. You let the traffic adapt itself to the lousy network design you've created. Lousy because it was against the traffic matrix, which is wrong for the ones actually going on. Okay. If you know the traffic, you design the network. It's going to be a lousy network for the real traffic. So you let the traffic have adaptive capabilities to match itself to the network it has to live on. That's called alternate routing. I recognize both and I put those into my work. And it says, *"Yes, do alternate routing because you've got to deal with imperfect networks."* Alternate, dynamic. And so, in fact, I discuss some alternate routing schemes in that book that you have some place -- my book. And then I did some more work at UCLA with a graduate student on some of these schemes that were referred to in the thing with O'Neal that's got Gary Foots, and I can get into that if you like.

Actually, I'd like a linear view.

Okay. My thesis was done. It was recognized for its technical contribution. Made into a book. My thesis has been made into a book. And I went off to Lincoln Lab to live my career. Okay? Except Lincoln Lab, who had supported me through all my graduate work, was an extremely liberal organization, and said, *"Look, before you commit yourself to working here, do yourself the favor to see what's out there."* And you know, they wanted me to work there, but they said, *"You should look around to make sure that's what you want to do."* So I got there in early '63, because I handed it in December '62, and I started interviewing some places. You don't really care about the interview process, but the point is UCLA made me this great offer. Bell Labs made me a great offer. I had done some teaching at MIT. Actually taught a full course on my own as part of my experience there and I just loved it. I never knew I'd like teaching that well. But I just loved it! And so I thought I'd try my hand at being a professor. It seemed to have a lot of the qualities I wanted in terms of my life and I'll list a few if you like separately. But the idea of what Lincoln Lab said and I agreed, *"Look, try it. If you don't like it, come back."* I tried it and the story -- I'm still there.

Right. Exactly.

The move to California, by the way, was a major move emotionally. My parents were on the East Coast. They hated me moving to Boston. You know, they're New York people, you know, and that's the world and that's where the family is. So

**moving there was hell, I thought. I took the bold move West, you know, It was really a frontier at that time in some sense, and I loved it. Okay.**

I want to get to your involvement with the Arpanet, with Larry and the Arpanet and when you first heard about it and all of that. Can we skip to that?

**Sure. So the initial _____ is Larry and I were good friends. He knew of my work. He respected it and we kept in close contact. In whatever year it was, I think it was probably '65 or so, he took a key position at Lincoln Lab running the group that was Clark and _____. And he came through one day and said he wanted to me with me because he had this option to go to this thing called Arpa.**

You had heard of Arpa?

**I had heard about it, but it wasn't all that dominant yet. They were not supporting me, for example. Another way I heard about Arpa was from Ivan Southerland, who was my other classmate. Ivan had the position Larry had for a small period of time, and he had initiated some work with UCLA trying to connect together three of the big computers on campus, which never happened for administrative and bureaucratic reasons. I was not coupled into that. That wasn't where I was going. So I met with Larry in Lexington. I remember we were sitting in his Volkswagen.**

What year was this?

**Either '65 -- what year did Larry go to Arpa? Do you know? Was it '65 or '66? It was a few months before he went.**

You were sitting in his Volkswagen?

**Yeah, in the streets of Lexington on a side street.**

He lived in Lexington?

**I forget where he lived, but he was working at Lincoln Lab which is in Lexington. That's where we met there. And he discussed the pros and cons. He wanted my advice. I gave him the same advice I took myself. I said, *"Look, Larry, try it. You can't lose. You can always go back to Lincoln. If it's exciting, go ahead and do it."***

Did he tell you about this whole thing about how they kind of twisted his arm?

**No. That's the way I heard later about they were putting pressure on the Lincoln support.**

And so you said, *"You can't lose."*

**Yeah. *"Give it a try. Be adventuresome."* And Larry certainly did not have an aversion to being adventuresome. He was an adventurer. And so he went there. He went there with this notion of wanting to set up this network for the resource sharing reasons. We began to talk about it early on. And then, you know, he was well aware of my work, and so in '67 when he wanted to lay down the framework for this network, he brought me and a few others back to Washington.**

During this conversation you said, did he say, *"They want me to design this network."* DO you remember any of that?

**No. I can't remember.**

See, because that is Taylor's whole thing of, you know, *"Well, we decided we would do this network. I knew the only guy who could build it was Larry Roberts. ..."*

**I don't remember. I can't fabricate that. I remember the emotional issue and not the technical issue.**

The emotional issue?

**Well, Larry's concern about making the move to Washington.**

I see.

**The career issue. But he indicated that he would have some important responsibilities there, so the issue may have come up.**

So anyway.

**So once Larry decided he wanted to do this, he looked to me for advice because I had laid down the principles of networking and he knew that I had done simulation. He needed to get some engineering principles down ahead of time to do some design for this network. So he brought a few of us back to Washington. I think you probably read this story in the Babbage Report. There were a handful of people. Anywhere from six to ten. I forget how many. But who was there. I remember**

**Tom Cheatum was there.**

In Washington?

**This was in Washington, yes. You know, I'm not sure where it was. I can picture the room. I think it was Washington. It was East Coast somewhere. It may have been in the Boston area, but I think it was Washington.**

What was the Gatlinburg?

**That was later. Gatlinburg was '72, wasn't it? This was not a Gatlinburg. This was not at a conference. This was a meeting.**

Oh, just for a meeting.

**A meeting to discuss how to proceed in getting this network created.**

So it was you --

**Myself, Tom Cheatum, Herb Baskin, I believe was there. It was a guy from IBM and I was looking up the Babbage Report a few weeks ago. Doug McElroy is the name that I have down there. I think that was the name. I'm not certain. Neat guy.**

[End of Side A.]
[Side B begins.]

Where were we? Oh, this is an important part. You sat down. They brought everyone to Washington. Larry --

**Oh, yes.**

When was this? Do you remember?

**Yes. 1967. But I don't remember the month. Our job was to discuss the concept of this network and to try to lay down some specifications, because he wanted to write the RFP to be sent out for bid. So we got involved just at this level of brainstorming.**

Do you remember any details of that?

Yes, I do. I remember two details in particular. It was the timesharing guy, who I think was Herb Baskin, but it may have been someone else, and he said, *"Look, whatever this network is, if it can't give me a half-second response time, I can't use it for timesharing."*

Right. This was the pounding on the table.

Yeah. And so we said, *"it shall be a half-second mean response time for short messages, interactive kinds of timesharing things."* And I said, *"Look, if this is an experiment, we've got to be able to measure the performance of this network,"* and I said, *"We've got to build in measurement hooks, software and possible hardware hooks."* So I specified the concept of measurement software; what some of the features would be. Be able to measure response time, buffer size, and all the rest. And so we said that would be part of the spec also. And we defined some notions of reliability. We laid down the concept of packet switching, described how it work and all the rest. The idea of alternate routing and adaptive control, distributor control, etc.

All going into the RFP?

I don't know how much appeared in the RFP, by the way. I don't think I ever looked at it. You should get a copy of it.

I've got it somewhere.

That would be nice to see, by the way. Because I'd love to glance at it, if you have it, so I could (print) some of it out.

Yeah, remind me.

Okay. So it went out sometime in '68 and a number of organizations bid, and I know at least one other organization bid the same computer, the Honeywell PDP-516, because one of my students was working for a company that did that. I found out later. I think it was CSC. I'm not sure.

Now, that's one thing that's very interesting to me is that a lot of people were using the Honeywell in their bid.

Yes.

So Honeywell was privy to a lot of what everybody else -- I mean, Honeywell had to please a lot of different masters, right?

**I'm not sure if these people like BB&N were asking Honeywell to be a subcontractor. If they were, the answer is yes. I'm sure they worked with these bidders. That's not uncommon. See, the machine had just come out in '68, as I recall. It was demonstrated in '68. The first time I went to the Spring Joint or Fall Joint Computer Conferences. And you know the story about the IMP #1, when it was on display? I believe we got one of the first machines that they'd ever manufactured. It was a military hardened machine.**

Oh, really. I'd love to see that. I've seen pictures.

**Well, it's at UCLA. Right there in the archives. So the spec went out. As you know, it was in January of '69. It was a BBN-1 and the die was cast.**

Let me ask you one more thing. Larry makes a big point that IBM sent in a no-bid, because they --

**The part that I know is they dropped out of the participation in this effort. Which makes me therefore think, Katie, that there was more than one meeting. Because if Doug McElroy dropped out, how did ____ dropped out?**

Right.

**So there must have been more than one meeting. I can't recall though. They were gone. I didn't know why. AT&T was not there.**

At the meeting.

**Right. So the biggest computer manufacturer and the biggest common carrier did not play in this game. And it's clear why the common carriers didn't.**

Yes, and now what Larry says IBM said was, *"There's no computer small enough."*

**I never heard that. To be switched?**

Right.

**Well, they missed the mini-market.**

Right. But you'd think they'd know that mini computers were being --

**Arrogance. By the way, Wes Clark was one of the first to build a mini computer. He built this little thing called the LINC.**

The LINC?

**Yes. Which had a little tape drive on it. Oh, it was a beautiful little machine. Anyway, AT&T was not there because they didn't believe there was any money in data transmission.**

Right.

**And they were arrogant just like IBM. But the reason IBM dropped out of the design build is because they wanted to bid presumably and they couldn't if they were in on the specification.**

Right.

**Because then they decided undoubtedly to go off in the SNA direction, which carried them very well through, you know, well over a decade, decade and a half. They made a lot of money in SNA.**

Anyway, BBN got the --

**They got the bid. So now maybe Larry can help check these dates, but let's talk about the Jerry Estrin thing, because he raised that. Here's where he comes in. If I'm not mistaken, some time around 1967, Jerry Estrin got a contract from, I think it was probably Ivan Southerland. That sounds a little bit late. Maybe it was '65. To get involved with this three-node network at UCLA. We had three major machines there -- 70-94's -- and they were the big machine at the time. There was one in the medical area, one in the western data process center, and one in the campus computer network, the general academic machine. Now you see, this is where the idea of networking -- the origins of (nepha) networking get fuzzy. Because I know that Ivan, who predated Larry, came by and said, *"Let's network those three machines because they're all the same architecture and it makes sense."* And Jerry Estrin got involved with that. He was the senior guy in the Computer Science Department at the time. He was not the Chairman, but he was there since the 50's actually. And he had built a computer in Israel; the first digital computer in Israel, the (White Zeock). He is a well-known figure. But it turns out, not very effective.**

Don't put that in, please. Because what happened is that project never happened for a lot of reasons -- bad leadership, bad bureaucracy, etc. However, Jerry still had this contract.

With?

Arpa.

Isn't that interesting.

Yes, he had a contract with Arpa. Now, in one of the Arpa PI meetings, and I think this was '69, but really I don't know how I'm going to go back and check. I probably could check. I was invited to the PI meeting, because they were thinking of making me a PI at the time, because the Arpanet was going to happen now and they wanted to support me to do this work with Larry on becoming the first node and doing the design, all the rest. At that PI meeting, Jerry Estrin, for whatever foolish reason, chose not to go. There was a conflict and he chose not to go, which was clearly a tragic mistake on his part. He sent one of his programmers up there, a guy named Vint Cerf, to describe some tools they were developing to do some measurement on a computer system. I forget what it was now. And they were talking about they were going to buy software. To be honest with you, Vint got totally decimated at that PI meeting. Couldn't defend what he was doing. They said, *"Your measurements are going to impact the system. You won't be measuring the same thing."* He had no answers. And the big cheese, Jerry, was not there. So it was a very, very bad piece of _____. As a result, Jerry got cut off. Arpa in those days was like that.

Really? They cut off the funding?

They cut off the -- well, they didn't renew. I forget exactly which phrase was used, but Jerry was no longer PI. Same time, I took over an Arpa contract. So it was essentially a transition from Jerry to me. Jerry had a group of programmers. Vint was among them. Steve was among them. John Postell was among them. They were running the Sigma-7 as well. So they had a small staff of people to support that. And I suddenly inherited all of those people to now refocus the entire direction to the upcoming Arpanet. And I grew that group from that smaller size up to a group of size 40, which included my graduate students. 40 people I was suddenly managing as a relatively young professor there.

You weren't even much older than they were?

Oh, no. My first graduate student was older than me.

No. Really?

My first PhD. The first class I taught at UCLA -- this is a kick -- they had this Extension Program where you could take courses at night and get daytime credit. So the first course I taught was one of these what's called a concurrent course. I taught at night. It was a course in communication theory. Almost all the students were older than me, because they were coming out of industry. Right? So I walked into this class with a crewcut. I don't know if you saw it. Oh, you didn't get the paper back. Is that my book, by the way? There's a dust jacket on it. The dust jacket has a picture of me. You wouldn't recognize me. My hair's all shaved off. Crewcut. So this kid walks into this classroom and starts teaching. And the first time, these tough guys there, they're going to basically challenge me. *"Who is the kid trying to teach us?"* So a particularly challenging question came up that first day -- my first class -- and I realized what was going on. This was a direct threat to me and my authority in that classroom. So the guy asked a question at this level and I answered it at this level, and I went bang, bang, bang, and basically hammered that guy to the ground.

Do you remember what it was?

No. It was some sophisticated question about communication theory. Because I was trying to teach them at a level -- you know, bring them up. And he was trying to tell me, *"Well, I know more than you."* So I went up two levels above him and I hammered him into the ground. I never had a problem with that class again. They were my buddies, you know. They listened to me. They respected me. So talking about age. Yeah. I had that right off. My first PhD student was older than me.

That's great.

At any rate, yeah, these guys were younger than I, but not much younger. And I formed, as I said, four groups out of that. I formed a hardware group with Mike Winfield in charge.

This was with the 40 people?

Yeah. Well, when I grew it to 40. I had a software group in which I put Steve Crocker in charge. And again, beneath him would be like John Postell and Steve Pinder, Charles Grime, and Vint Cerf, and some others. I had this Operations

Group. Lou Nelson was in charge of that. Running the machine and the staff. And then my Theory Group, my Design and Analysis Group, which was me and my students.

The reason you inherited these people was that you became the PI?

I became the PI and I was given the job of making the first IMP host connection.

And that's why they all moved over?

Yes. Yes, I needed a staff and I took them on, because I needed to get the mass of programmers and hardware designers and theoticians and machines running. It was natural. You know, you don't want to fire these people. They were there as resources. Made sense.

So at the same time the contract was awarded to BBN, you got --

I think it was April of '69 when I got my support, which was just shortly thereafter.

And how much money was that?

I can't tell you. I can tell you ballpark. It was probably on the order of half a million a year. A good million not too long after that. Because there was equipment and travel and mostly staff, you know, students and staff. A small amount of summer money for me. I was very cheap, you know. All they did was pay my three months in the summer. That's all I could earn. During the year, I'm free to the contract. That's how UCLA works.

Oh, really?

Doesn't cost them a penny. It's not like a place like MIT or USC where you have to support half your salary. When you get a position at UCLA, you're paid nine months. Three months in the summer is up to you. That's where the contract had been. Anyway, so we got rolling. I got into the East Coast. I assigned tasks to each of them. The main focus was the host IMP connection, first of all, locally. But secondly, supporting the entire network to start laying out the design principles and the tools to evaluate the network and provide topological design tools to offer to grow the network, and to discuss the routing procedure, because this routing was very important. This distributing control routing procedure which I had laid out in my book and now we had no one working. So I got Gary Phelps working on this and

we developed -- we knew that BBN was planning to implement something called periodic update. The algorithm that they developed based on things that I done in the mid-60's and in my thesis was an (alvin) algorithm where nodes exchange information about what they saw. So for example, you tell me that you've got a good route to some destination, and this neighbor tells me that he's got a good route. Well, if he tells me better than you, I'll use him. And based on what you tell me, I'll take the best and I'll tell my neighbors, including you, what my best routes are.

What is that kind of routing called?

It's called distributed control. Okay. Where nobody's in charge. There's no guy on the top that says, *"Okay, the route shall be this."* Everybody's passing information around.

Was this a new way?

Well, yes and no. Some of the ideas were present way back. There's a guy named Reece Prosser, who talked about -- the mathematician I quote in my dissertation -- talked about routing. Paul Baron had been talking about hot potato routing.

On, now Larry has said that he took the hot potato routing from Paul.

Yeah. But that was a very elementary version. Hot potato routing is I send you a message. If it's not for you, you send it some place else.

Right. You mean you just throw it?

Throw it. Throw it with no direction at all. That's called random routing.

Where does it go?

Well, I analyzed a random routing procedure in my dissertation as well. Because Reece Prosser had talked about it as well. And I looked at random routing and I found the way -- how poor and good it is in different situations. In fact, what's interesting there is I did some simulation -- some analytic results -- and I got a beautiful analytic result, a very simple answer, and I took it to Shannon and I said, *"Why is this answer so simple for this complicated problem?"* He thought about it a few minutes and he came up with the physical reasoning why. The man was just brilliant. I mean. And he hadn't been working it. I just came to him with this and I used him as advisor. Bang! It came right out.

Wow. I want to understand this routing. Did BBN have to say in its bid what kind of routing it was going to use?

**We gave them the idea for the routing.**

And what was the idea you gave them?

**You have the RFP? It would be so important if you have that.**

[Tape is turned off.]

Okay. It was specified in the RFP.

**The RFP called for periodic updates of the routing messages. And the measurements specifications were also in the RFP that I had created, as well as the model of the network and the performance measures. Yeah.**

So not this hot potato stuff?

**No. No. Hot potato was not the way it was done. It was based on evaluation as to good routes that each of these IMPs would pass to their neighbors.**

So I'm the message. I need to go somewhere and this IMP says, *"I've got a good route."* This IMP says, *"I've got a better route."*

**Well, not quite. It's done before the messages necessarily arrive. I have to get a routing table. A routing table means when a message comes to me headed for some destination host, I've got to just decide what next hop to make for that message or that packet. So I've got to keep a table, because it it's going to MIT's PDP-10, go from here to Utah, IMP, and Utah will decide what to do with them next. There's no total path notion. How do I get that information? Well, the neighbors tell me. They bid and they tell me what they think is their best -- the minimum delay. So suppose I can believe my neighbors. Okay? I take those numbers and I look at my delay to them. You give me a number. 30 milliseconds. I got 10 milliseconds to you. I'll add that. Ba-ba-ba. I look at all those numbers. I take the best. The best is the guy I'm going to name as my next hop. But every so often, roughly every two-thirds of a second, we exchange messages again, and maybe things have changed. You have a longer queue now and a shorter queue here, whatever. Well, that's where the dynamic comes in. Length of the queues. And so we're only doing local exchange of information, but carrying global information.**

I see.

**Okay. By the way, we also specified, you noticed, a host to host traffic flow. You know how that came about? Before the network was implemented and we knew it was going to happen, I called up every host in the network and I said, *"How much traffic will you send from you to each other node?"***

Oh, right. But how many hosts were there?

**Not many. I think I called -- if we find my paper, there were probably between 10 and 15 hosts. A host was one computer at one site typically.**

There's no way they can give you any kind of -- anything that's going to be an accurate figure.

**No way. Well, they said that. They said, *"I have no idea."* I said, *"Well, will you send like three teletypes worth of traffic to the Iliac?" "Yes. Three teletype traffic." "And how much will you let the network use you?" "I don't know." "Well, how many teletypes worth?" "Four teletypes worth."* So I took those numbers and I published them in a table and used that for my model, and that resulted in a couple of papers which describe the performance of the network. In fact, these are the documents I told you about earlier. This came out of the Spring Joint Computer Conference of '70 and of '72, and these are the five papers. This was one session -- was five papers. And I'm sure you've seen these papers. But there's all _____. I can't give you this. It's the only copy I have. But for example, I have here that particular network, which was a hypothesized 19-node network in the Spring of '69. That's UCLA right there. One of these is MIT. And from that, able to get some performance characteristics as to the way the network would behave. In fact, already -- now this was Spring of '70, okay? Already I am showing fixed routing and asynchronous -- this is fixed driving, which means periodic updating. This is a periodic updating with and without loops being suppressed, the performance. So I already had all of that laid out for them to build their algorithm on. And in fact, we had invented it. After the periodic updating was specified, as you see here, I was looking at asynchronous updating, which meant the following. Okay? In the periodic, every two-thirds -- every 640 milliseconds, two-thirds of a second, you send me a new update as to what your view is.**

What do you mean *your view*?

**What your minimum delay to every other node in the network is.**

And that's every?

**Two-thirds of a second.**

My goodness. How does that go so quickly?

**Well, the lines are fast. 50 kilosecond lines. The messages were moderate length. But it was traffic in the network. In the early days, it was the only traffic in the network, because there was no real traffic going on, that and the measurement traffic. I'll tell you about that in a minute. But my thought was, look, if two-thirds of a second comes by and you've got nothing new to tell me, why bother me? And if you just told me one thing and suddenly a major change takes place, a line breaks or you receive a huge load of traffic, why not tell me immediately? So tell me when you need to and don't bother me when you don't. That's the notion of asynchronous updating. And that performed far better in our simulations and on our theory. And so I told Frank about that, and you know what his response was.**

Frank Heart?

**Yeah.**

What did you tell him?

**I said, *"Look, two ____ updating is fine, but asynchronous updating is better."* His response was, *"It's working fine. Leave me alone."***

When was this?

**I can't tell you exactly. It's shortly after the network was implemented. Very early days. And he probably gave me the right answer in some sense, because it was not a crucial thing at that point.**

Well, how hard would it have been to change it?

**At that point, I have no way of knowing, because we didn't have the code. Nothing is trivial in software, so it would have taken some effort. But the fact is in 1979 they did implement that kind of _____. Took them 10 years to do the damn thing.**

They did do what?

They implemented an asynchronous type of updating algorithm.

Oh, really?

Yeah. With some other features, but that was the basic components. We had shown them years before.

Who is *they* at this point?

BBN was the one who implemented this stuff. It was called the new routing algorithm at the time. I think that John McQuillen was involved in some way.

I was going to say, because I went and talked to McQuillen and that sounds familiar.

So where are we in the story now? I'll repeat what I said at lunch. I think that the truth that this adaptive distributed control routing algorithm worked was probably the biggest technical success of the Arpanet or the Arpanet experiment in those early days, because people did not believe it could work. We described it and they said, *"That can't work. There's nobody in control. It's going to go haywire."* And yet we showed analytically and in my simulation that it would work and the ultimate truth was to build the system and show it to the world. However, the particular way in which I saw Frank was implementing this periodic routing algorithm, we could see early on, because we looked at it mathematically and simulated it. We could show there would be loops. Then we get to the loop _____. Okay?

You could see by the way Frank was implementing --

The algorithm that it would permit loops in the network. A loop means -- a simple loop is I send it to you, you send it back to me, I send it to you. It's caught in a trap. Or A, B, C, A, B, C or bigger ones.

How do loops happen?

I'll explain that in a minute if you like. But let me tell you the effect of it. So we called up Frank and said, *"Frank, you got loops in your routing algorithm."* Not saying that it's a bad thing. Just *"Frank, you got loops and it's going to cause some delays."* He said, *"We have no loops in the algorithm."* I said, *"You've got loops in your algorithm."* He said, *"We have no loops in the algorithm."* This was before the network was built. Okay? I told him that. *"No loops."* You know, you can only beat on a guy so long. We had calculated analytically what happened. We simulated

it. It happened. We implemented the network and we showed them loops. Measured loops. You can't argue with that. His answer was, *"Yeah, you get loops. But the loops go away."* And they do and that's okay. The loops go away because the algorithm is smart enough to recognize when loops occur to change the path. Because if there is a loop, it's taking me too long to get there. Right? And the clever algorithm knows that. Now there is a philosophy here though. You might say, *"Let us legislate against loops."* That's a bad idea. That'll make the network crash.
Why does that make the network crash?

Where's your --

Notebook. It's right here.

Yeah. You showed me a picture of some networks a little while ago in one of the pages, I guess. Oh, here they are. Here you can see the PDP-10's. This was the kind of picture I was going to show you. Look around. Look at all the PDP-10's. PDP-11's were little serving machines. So they tried to knock down _____. These are old switchers. Okay? Suppose you got a message to move and the packet is moving down here. Okay? Suppose that IMP crashes or that line crashes. Packets arrived here. The only way to get to here is to go back around. So you've gone boom, boom. That's a loop. If you say it's not allowed, the traffic will die.

I see.

See, you can't legislate against them. Now see the way you said, *"I see."* Well, that's the way deadlocks happen. People don't see until you show them and it's very hard to anticipate some of those things. Now that's an easy one. But the thing about doubling the storage, that was an implementation problem. We found some very sophisticated deadlocks. They're almost impossible to predict, but they're real easy to fix once you see them. Okay. So now where are we?

You tell me. When did you tell Frank about the loops?

That was before the network was implemented. Sometime in '69, and it took 10 years before they implemented that. But to be fair, the periodic updated worked okay. And that's an example of where I say they were not interested in ultimate performance. Dave admitted that programmers -- not necessarily his group, but typically, only worry about getting the program to run. How fast? That's somebody else's problem. Please, God, let the logic be correct so it doesn't crash. That's

**enough of a programmer malady. Well, programmers are more sophisticated these days and they worry about performance, but in the early days they didn't.**

No. Crowther did a lot of the programming, didn't he?

**Crowther did a lot of programming, yes.**

Is he good?

**Oh, he was a genius. He was great. The key guys there were Walden, Crowther, Severo Ornstein. I think, those are the ones I could see really carrying it off.**

Really?

**Yeah. They were the engineers down in the bowels of the system. That's the BBN side. On our side, the key players in my group, Michael Winfield, Steve Crocker. Those are the two keys guys. And I don't want to bad mouth Vint, but he did not have an important role to play in those early days. He was there. He was doing his job as a programmer. But it wasn't key. His role came when he got to the TC _____. That's when he got to do something significant that was head and shoulders above other people.**

Right.

**And that's why having him called the father of the Internet bothers me, because he's not the father of the Internet. He's being heralded as *the*, not *a*, but *the* father of the Internet. It's wrong. But don't quote me on that. That's just to put in your head.**

Right. Okay. Now where were we?

**In fact, Vint wasn't even there when we sent the first message between two hosts.**

Oh, he wasn't?

**He was not present at that event. At least so far as my guys recollect. That's not my own recollection, but my guys recollect that Charlie Kline and Johnny Barsted were there, who sent that first message back and forth -- the SRI host.**

Well, this might be a good time to step back and any details you can remember about that first day. I think we went a little too quickly over that first day when everybody was

there and nervous.

**Yeah. But you know, we connected it up. The bits flowed. We were all pleased. And that was pretty much the end of it. There was nothing very -- the next day we sent some messages. *Send bits* means we looked on the lines and we saw digital bits moving back and forth, and they were getting from one --**

[Someone comes in. Tape is turned off.]

**It was like any other day. We got the bits moving back. Next day we got full messages moving from host to IMP and we could loop it back and see. We could talk to ourselves.**

So you were just going host to IMP because there was no other --

**There was no other host. There was not other IMP. A month later, we had another IMP and another host and then we sent a host to host message.**

To SRI?

**Yes. And then Santa Barbara.**

Is that a bigger deal?

**In terms of our perception and making the day an event, no. No. I had Charlie Kline send a message up to them. Just a logging message. And they were communicating with a piece of the channel which we use as a voice telephone line with a multiplex voice. They had headsets on. Charlie did and the guy at the other end, who name I forget right now. And they said, *"Did you get the message?"* And he says, *"Yes and no."* You know, we needed that aid to make sure that it was happening. So the first message between hosts happened in early October. I've been asked this question by a lot of different people.**

What was the first message?

**Not only what the first message was, but did you realize at that time? And the answer is we knew we were doing a good engineering job. You know, this was a major success in that sense. But did we know we were changing society? No way. Did we know we were changing the way computers would talk to each other? Absolutely. Because they couldn't talk to each other before this. But you know,**

early 80's, which was 15 years after we proved the working technology in a working prototype. In 1982 or '83, I forget exactly, when AT&T came out with their premier networking service, Net 1000, three years later in '86, they closed it down at the cost of a million dollars. AT&T couldn't make it happen years and years later for a variety of reasons.

Matt: Yeah. You mention that in here.

Is it in there too?

Matt: Yeah.

Okay.

Matt: It seems to me that they all just missed the boat. Is that -- I mean --

Their failure at that time was not a technical one. It was a marketing one. It was to whom they were trying to build this network and what they were offering. They were trying to sell computing and storage and offering a programming language to the Fortune 500 as well as networking. The Fortune 500 doesn't have to buy computing cycles from AT&T or storage or Cobol which is what they were offering the customer, Cobol. So there was a major mistake in what they were trying to sell and it was late. Meanwhile, these other organizations were up there selling service. The Telenets and Timenet and Uninet and a bunch of others at that point.

Matt: You said in that interview what AT&T didn't understand was that all you needed was 100 milliseconds of time to transmit data. You weren't looking for that three minutes that they were -- the increment they were charging for.

Oh, that was early on.

Matt: That was early.

I don't think I talked about that in regard to their 1983 network. This was pre-Arpanet. That's right. That was one of the big issues. In those days, AT&T said, *"Use the telephone network for your data transmission."* And my point is three minutes was outrageous in terms of delay. Three minutes was the cost issue. Many seconds to set up the connection. You've got to dial and wait until it answers. When

you want to send 100 milliseconds of data, it was outrageous. You're not going to wait tens of thousands of milliseconds to send 100 milliseconds of data. So they had no clue as to what the issue was, A. B. They saw no market there. And in some sense, in those days, they were right. Voice was the market, not data. And even today the thing that's driving on digital data network is digitized voice and fax now, by the way. Fax is a big revenue source for the telephone company. While Katie's out there, I'm going to make sure. I know it's precious time.

[Tape is turned off.]

Matt: Frank sort of says the loops don't really matter.

But you've got to see why we were saying those things. Franks job -- he had a contract to deliver a network and continue to operate it. What he didn't appreciate or wasn't in his head -- I don't thing it ever could get there -- this was not delivering a stapler to be used in production. This was an experiment on what is networking about. Let's probe it. Let's develop this technology. Understand the antithesis here. They really are opposing views. And that's the way it was. I mean, this was the grand experiment! We recognize that now. Where if Frank had his way, we'd still have those mini computer still operating the god-damned thing with no improvements at all. You know, and he would have been happy as a clam. *"Look, it's working!"*

Matt: Well, that raises a -- maybe it's too general a question for us at this moment, but it raises a question in my mind that I certainly see in a lot of this discussion. The issue there seem to be various ways of meeting major objectives. Everybody sort of has a different way of approaching them. Sometimes you could do it this way. Sometimes you could do it the other way. The effect is the same.

I think Dave Walden in his reply here had a good way to say it. From his view, and he's correct. We had a good relationship with people like Dave. It was Frank with a problem. This was a cooperative collaborative effort, and if we don't agree, we'll still go ahead. You know, we can agree to disagree. Let's move ahead with things. That's what made this thing happen. Frank was holding back on the reins.

Now you were going to put something on the list?

Yes. I want to put a few things on the list.

Before you forget.

**Okay. What is on your list right now?**

Okay. We got Estrin, _____. Oh, AT&T's lack of interest, Howie Frank, the story with Frank and looping we got.

**I want to tell you about Howie Frank and how I met him and how I had him meet Larry. But then the other things were I want to talk about the Arpa satellite packet switching and the ground radio packet switching and how that led to lands. Then jumping way forward, the 1988 report which led to gigabit networking, and this 1994 report, just realizing the _____ future. Now you may never get to that. But those are some of the things. Oh, another thing is the army of students that came out of my work.**

Oh, yes. Right.

**That's an important.**

Matt: Yeah. The _____ you refer to.

**Yes.**

Matt: I think that's really quite interesting.

Right. And Alex McKenzie, by the way --

[Tape is turned off.]

**Well, we just got through the first and second days in the first month.**

We went to SRI and back.

**Right.**

And that went without a hitch?

**Yeah. Well, yeah, basically without a hitch. Now let's say a word about use of the network. What was happening on the network at that time. Did you finish your list, by the way, Matt?**

Matt: No. But that's okay. Keep going.

So this four-node net got installed by the end of December and we started looking at the inner workings of the network and how well it was performing. That was fun. Except there was no easy way for a user on one host to use a host across the network, because there was no host/host protocol. And that job was assigned to me and I gave it to my software guys to develop the host/host protocol, and that was supposed to be ready the summer of '90. Okay? Six month project. Didn't really get done until the summer of '91. It was a way delay because we really didn't anticipate, A., how complicated that problem was.

'90?

'70. Sorry. '70 and '71. I'm sorry. We didn't appreciate how hard the problem was and also we gave it to a bunch of graduate students to do. This was not something you give to a Frank Heart to do. And so as graduate students are, they take their time. They work as a distributed group. They do a lot of traveling. In fact, that's a little anecdote there. Ever so often, Steve would come to me with his software gang and beat on me for more money. Usually they wanted to go to some exotic place to have a meeting. And I would say, *"What are you talking about?"* They'd say, *"Yes, we got to go."* More often than not, I'd say yes to them, *"Do it."* So they took liberties. But you've got to understand these guys were in some sense uncontrollable. You know, they were creative and independent and you couldn't manage them with a hard hand. It would be a big mistake to do that. You had to let them --

Matt: Where did they meet?

Oh, certainly in Alta a couple of times. I think Hawaii once. Probably up in Utah a few times. Because the Utah people were helping quite a bit. These graduate students of that ___ all around. I was reminded of that because the use of the network, in the early day, there was no host/host protocol. You didn't know how to address another host. So what was going on the network? Measurement traffic, routing traffic, and most significantly, these researchers, be they graduate students or faculty, some of them would migrate from one site to another. Particularly, there's a fellow whose name is in the Babbage Report, who moved from Utah to Palo Alto Research Center, and he knew how to use the machine in Utah, of course. So he knew exactly how to get through the network and what the command structure was and the lock____ position, all the rest, and he wanted to use that machine because he couldn't use the machine locally as easily. Okay? It was a foreign machine to him. So that kind of use was the main use. People who migrated and wanted to use the old machine. Plus a few one on one things. But by the time '91

**came by --**

Matt: '71.

**'71. Thank you. The host/host protocol was established. We began to get connected and then it was a much more simple way to use _____.**

You mean, it was basically useless?

**No. Unused. Difference. There was not much going on. Again, don't forget the reluctance on the part of these hosts to be attached.**

Matt: That's very interesting to me and that was kind of one of the things --

**Well, here was the argument. I'm a PI at say Columbia. I have a machine. One day Arpa comes to me and says, *"Connect your computer to the network."* I say, *"Why?" "Well, so other people can use it." "What? It's loaded up to 100% as it is. Go away." "By the way, you can use other people's machines." "No, no, no. I'm not going to use it."* That was the argument. Okay? That was real.**

Matt: And we're talking about resource sharing at that point as opposed to --

**Resource sharing was the prime mover in those days. And so they realized people would use their resources.**

Resource sharing means that you log onto someone else's computer and use like their graphics capability?

**You would use it right there. Use the application there. You don't download it.**

You're not taking data or anything?

**Data as well, but that was not seen as the main application.**

Matt: And Larry Roberts was instrumental in overcoming that resistance of the hosts?

**He and I. I was the one who called them all. He also helped.**

Oh, you called them all?

**That's where I got my traffic makings from when I was talking to them. I said, *"Are you going to connect?"***

Oh, right.

Matt: And what did you say to them?

**I explained the benefits of sharing. That this was a grand experiment they should try. But Larry put the arm on them as well. He was splitting them. So he really had the clout.**

He was giving them the money. Now I know how those first four nodes were chosen. How about those 15?

**I have no idea, except we wanted to go across country very quickly, and I believe the first link was either to MIT or to Harvard, and that was in the summer of '70. We were already across. And it's in my damn pictures here. My volume two which we don't have here. But I think they've got the key pictures right here. I think they got this from my book. I'm pretty sure I gave them to them. They got a lot of these from me, by the way. I remember I gave them to Judy. Yeah. By June '70, we had two links across the country you see, MIT and BBN. And I don't know what the criteria was for selecting them.**

**But this brings us to the Howie Frank story actually. Howie Frank did his PhD sometime in -- he finished in the mid-60's, probably '63 or '64, I'm not sure exactly when. But he tells the story, as he did at the Arpa PI meeting the other day, when he got up there and said -- he related the anecdote again. He says when he was a graduate student working on his PhD, one day someone told him that someone has already published his dissertation.**

Oh, right. I know this.

**Okay. So I'll skip the story.**

I've got it. But it's a very amusing story.

**Yeah. To Howie's credit, he continues to tell that story.**

As if it's the first time he ever told it.

Yeah. He's a good story teller. So he knew my work, and then he asked me -- he was running a short course at Berkeley. He was Assistant Professor at Berkeley. And he asked me to participate in it. And I said, *"Sure."* Summer course. And it came at just the time when I was going to be with my family and four kids camping in a tent up in Sequoia National Park. I said, *"Fine. It's a one day thing. I'll get up early in the morning. I'll drive down to Fresno. Catch a plane up to Berkeley. Come back and get back in one day."* Why not. Mainly, I wanted the money.

[End of tape.]